SRI International

THE GHOUGH GENERALIZED HOUGH TRANSFORM PACKAGE: DESCRIPTION AND EVALUATION

Technical Note No. 288

December 1982

By: Kenneth I. Laws, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Program Development by:

Dana H. Ballard
Kenneth R. Sloan, Jr.
Bill Lampeter

University of Rochester

SRI Project 1009

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-79-C-0588.

SRI International®

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1982		**00-12-1982 to 00-12-1982**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Ghough Generalized Hough Transform Package: Description and Evaluation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**49**	

Foreword

The primary purpose of the Image Understanding (IU) Testbed is to provide a means for transferring technology from the DARPA-sponsored IU research program to DMA and to other organizations in the defense community.

The approach taken to achieve this purpose has two components:

(1) The establishment of a uniform environment as compatible as practical with the environments of research centers at universities participating in the IU research program. Thus, organizations obtaining copies of the Testbed can receive a continuing flow of new results derived from on-going research.

(2) The acquisition, integration, testing, and evaluation of selected scene analysis techniques that represent mature examples of generic areas of research activity. These contributions from participants in the IU research program will allow organizations with Testbed copies to begin the immediate exploration of applications of IU technology to problems in automated cartography and other areas of scene analysis.

The IU Testbed project was carried out under DARPA contract No. MDA903-79-C-0599. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

This report describes the GHOUGH generalized Hough transform package contributed by the University of Rochester and provides an evaluation of its features.

Andrew J. Hanson
Testbed Coordinator
Artificial Intelligence Center
SRI International

Abstract

GHOUGH is a computer program for detecting instances of a given shape within an image. It may be used for cueing, counting, or mensuration. GHOUGH can find instances that are displaced, rescaled, rotated, or incomplete relative to the shape template. They are detected by computing a "generalized Hough transform" of the image edge elements. Each edge element votes for all those instances of the shape that could contain it; the votes are tallied and the best supported instances are reported as likely matches.

GHOUGH was contributed to the DARPA Image Understanding Testbed at SRI by the University of Rochester. This report summarizes applications for which GHOUGH is suited, the history and nature of the algorithm, details of the Testbed implementation, the manner in which GHOUGH is invoked and controlled, the types of results that can be expected, and suggestions for further development. The scientific contributions of this technical note are the analysis of GHOUGH's parameter settings and performance characteristics.

Table of Contents

Section 1

Introduction

GHOUGH is a computer program for detecting instances of a given shape within an image. The instances may be displaced, rescaled, rotated, or incomplete relative to the shape template. Shape instances are detected by computing a "generalized Hough transform" of the image edge elements. Each edge element votes for all those instances of the shape that could contain the observed edge. The votes are tallied and the best supported instances are reported as likely matches.

The package was originally written in the SAIL language by Drs. Dana H. Ballard and Kenneth R. Sloan, Jr., at the University of Rochester. It was rewritten in C and prepared for delivery to the IU Testbed by Bill Lampeter at Rochester. At SRI the code was integrated with utility and image display software derived from other contributions, particularly those from Carnegie-Mellon University. This integration was done by Dr. Kenneth I. Laws.

Numerous changes were made in the user interface to facilitate evaluation activities. A few extensions were made relating to multiple target detections, but the central algorithm has been left virtually unchanged. The information in this document should thus be considered supplementary to the material cited in the references.

This document includes both a user's guide to the GHOUGH generalized Hough transform package and an evaluation of the algorithm. Section 2 explains the nature and uses of the algorithm and suggests alternate approaches to similar data analysis problems. Section 3 surveys the historical development of generalized Hough techniques and presents the current algorithm in detail. Section 4 instructs the user in the mechanics of using the GHOUGH software. Section 5 goes into greater detail on the meaning of the user-specified parameters, documents the performance that may be expected in various circumstances, and presents the results of evaluation tests. Section 6 outlines a number of suggestions for improving the algorithm and its implementation.

Section 2

Background

This section presents a management view of the GHOUGH program. The template matching algorithm is briefly sketched. Typical applications and potential applications requiring further development of the algorithm are discussed, and related applications for which other algorithms are better suited are noted.

2.1. Method

GHOUGH is a program for detecting instances of a given shape within a digital image. The shape is described by a template -- a series of dots outlining the shape and forming a closed boundary. Only silhouette information is used: this provides noise immunity in many applications, but does limit the use of GHOUGH in infrared target identification and other low-resolution applications.

Templates are abstract shapes, without size or orientation. They are also stored without contrast information so that instances may be located against darker or lighter backgrounds. (For efficiency the GHOUGH program allows the user to restrict the search to particular sizes, orientations, contrast directions, or subareas in the image. Shape instances outside these ranges will not be found.)

The GHOUGH program passes an edge detector over the image. The edge detector computes both a gradient magnitude and a gradient direction at each point (pixel) in the image. The user must set a threshold value for choosing which gradients are strong enough to be considered valid edges.

Each edge is allowed to vote for all shape instances of which it could be a part. One vote is recorded for each point in the template at each permissible orientation and size. This vote corresponds to the computed (x, y) coordinate position of the shape center. (The shape position is indicated by an arbitrary reference point in the template. Typically this point is at the center of the shape, but for a corner template the logical position would be the vertex of the corner angle.) Two separate votes are recorded if the shape may be on either the dark or the light side of the detected edge. The votes are added into an accumulator array consisting of one cell for each permissible shape instance.

Each cell of the accumulator corresponds to an (x, y) coordinate position in the image at which a shape center might be found. Each cell also has an associated rotation and radius, so that there may be many cells corresponding to each (x, y) position. The set of cells belonging to a single (rotation, radius) pair is called an accumulator plane. For efficiency the search may be limited to template center points spaced every few pixels in each image direction and to rotation and radius values that are also coarsely quantized.

Each edge votes for many potential instances of the target shape. Some cells in the accumulator will receive only a few votes while others will receive many. A cell

corresponding to a true instance of the shape should receive far more votes than other cells around it. Shape instances may thus be found as local maxima in the accumulator. The few maxima receiving the highest counts are reported as likely matches.

2.2. Typical Applications

The GHOUGH program may be used in any application where a known shape is to be found or measured in an image. It will detect instances of the target shape within a specified range of sizes or orientations and can do so in the presence of noise, occlusion, weak shadows, and some types of texture or camouflage. Matching is done in two dimensions, without regard for perspective changes or image warps. Large shapes are located more reliably than small ones, and high-contrast instances more reliably than those that match their backgrounds.

The program can detect or verify the presence of expected image features as in registration, terminal guidance, and cueing of specific targets. It is well suited for coarse registration (finding the first landmark) and for fine registration of individual landmarks, but not for simultaneously finding many different landmarks.

Once a target has been located, the program may be used for camera calibration or change detection. These applications require very fine quantization in the accumulator to accurately determine the rotation angle and scale of the landmark image. Camera position can then be derived if the landmark dimensions are known. Such information is useful in aerial image registration and in navigation.

Parameters measured from a calibrated imaging system may be used for image analysis and change detection. The GHOUGH program is only capable of finding matches to expected shapes. Analysis of shape changes, as in snow cover or lake boundaries, requires higher-level techniques. Better tools for analyzing close perspective views of three-dimensional objects may be found in the ACRONYM system [Binford81] and in various warp-registration systems [Chien75, Beisher79, Clark79, Lowe80, Noges80].

Temporal change detection and scene analysis problems often require that irregularly shaped regions of one image be identified as corresponding to regions in the other image despite contrast and location differences. This can be done by segmenting both images and establishing region correspondences [Price75], or by segmenting one image and using the GHOUGH program to find the same regions in the other image.

Tracking of large objects, as in weaponry fire control, is an application of image-to-image registration for which the GHOUGH program is well-suited. The shape can be extracted from the first image frame (or from a target database) and can be used to track the object through succeeding frames. Some augmentation of the GHOUGH algorithm is needed if perspective changes of three-dimensional objects are to be accommodated during tracking. It should be noted that tracking is a well-developed area of image processing and that many special-purpose algorithms already exist [Flachs78, WDESC78, Woolfson78, Choate79, Dougherty79, Fitts79, Flachs79, Goodman79, Maybeck79, Orton79, Pridgen79, Reid79].

The GHOUGH program is ideally suited for finding multiple instances of a shape, as in industrial bin picking, microscopic particle counting, or railroad yard monitoring. It might find use in counting the buildings or finding all the right-angled corners in a

high-resolution urban scene. Other straightforward applications are character recognition (with a limited symbol set), industrial parts identification, and sensing the position of a robot arm. It could also be used to search for additional parts of an object or structure once a subpart has been found. Other edge-based template matching techniques [Hemami70, McKee76, Perkins76, Bolles82] may be competitive for all these uses.

A particular example of shape detection is the identification of circular structures in an image. Circles in aerial images often indicate storage tanks, while those in ground-based imagery may indicate vehicle wheels, storage tanks, or other man-made artifacts. The GHOUGH program can be used to identify all circles or circular segments in an image (within a specified range of radii and contrast parameters).

GHOUGH can also be used to find ellipses or parabolas of any orientation, although the aspect ratio of the shape must be fixed. Ellipse detection is used to find circular objects viewed from an oblique angle or cylindrical parts scanned by a light beam [Bolles81]. Hough parabola detection has been used for locating ribs in chest x-rays [Wechsler77]. More general techniques for finding parametric curves of unknown aspect ratio will be described later.

Texture analysis is the identification of a texture pattern or of the process that produced it. Some methods of texture analysis require that subpatterns (*i.e.*, texture elements, or "texels") be identified, as in the identification of an orchard by the regular spacing of the trees. The GHOUGH program is well suited to finding the texture primitives providing that they are all similar and have sharp outlines. The program cannot address the harder problem of knowing which texture elements to look for in the first place.

2.3. Potential Extensions

The GHOUGH algorithm is essentially a pattern recognition technique [Ballard81a]. Minor modifications to the data collection processes will change the class of patterns to be detected. For instance, a line detector could be substituted for the edge detector in order to locate thin rivers, roads, or other linear features in an image. One such approach that uses line segments instead of points or edges is described in [Davis82].

The current restriction that templates be specified as closed curves is not inherent in the algorithm. Future versions of GHOUGH may permit linear or arbitrarily connected templates. This would increase the reliability of detecting linear features and common shape fragments (*e.g.*, object corners). It would also permit internal detail to be used in a template as well as silhouette information.

A particular case of linear feature detection is the detection of all straight line segments in a scene. This is especially important in navigational obstacle avoidance, terminal guidance, and robotic manipulation. The Hough algorithms grew out of such applications [Hough62, Griffith70, O'Gorman73, Perkins73, Doudani77], and the GHOUGH program could easily be adapted to this purpose. Lines segments or wires found in a scene would have to be verified by other means since the Hough transform is not well suited to finding the end points of a segment.

Circle detection has been discussed above. Although the current program can find circles and other analytic shapes, it is not as proficient at these tasks as a special-purpose algorithm. If circle finding is important, it may be advantageous to use a

special accumulator for the purpose. This is very similar to the adaptation for line finding mentioned above. Hough curve detectors have been studied by several authors [Duda72, Merlin75, Kimme75, Ballard76, Tsuji77, Tsuji78, Sklansky78, Shapiro78a, Shapiro78b, Shapiro78c]. Least-squares fitting [Bolles81] and other pattern recognition methods have also been studied, and may offer advantages over Hough techniques. Once a curve or boundary has been located, Hough techniques offer one of many approaches to segmenting the curve into meaningful fragments [Shapiro79b].

Tracking point targets is easier than area-based change detection. Hough techniques can be used to cluster motion or optic flow vectors to find groups of objects moving in formation [O'Rourke81b]. Other generalized Hough techniques have been used to track changes in line drawings or edge maps and to check multitarget trajectories for consistency [Yam81].

Hough methods can also be used for finding vanishing points in perspective scenes and foci of expansion in optic flow images. A method has been developed [Ballard81b] that uses a Hough line finder on the original image and a similar circle finder on the accumulator. This works because all lines through a vanishing point in the original image will cluster along a circle through the origin in Hough space. It may be possible to develop a single-stage Hough-style transform to replace this two-stage process.

Several Hough techniques are candidates for detecting complex composite shapes. Articulated shapes, such as the arm of a robot manipulator, are best found by locating the individual parts and then using higher-level knowledge to recognize the composite. Complex rigid shapes, such as a vehicle, may be found by a similar technique or by convolution in the accumulator [Ballard81a]. The convolution technique is based on the fact that an edge element votes more accurately for nearby reference centers than for template centers further away. One can therefore increase recognition accuracy by breaking the template into smaller chunks, voting for the chunks independently, and then combining the votes according to the template decomposition. This permits recognition in cases where individual subparts may not be identifiable without contextual information. Another approach, called the hierarchical generalized Hough transform, has been developed by Davis and Yam [Davis82].

Several researchers [Ballard81b, Ballard81c, Hinton81, Sabbah81] are now working on generalizations and implementations of the Hough transform for intrinsic image estimation. This is the conversion of an image, or a sequence of images, into explicit maps of edge locations, color regions, surface orientations, distances from the viewer, optic flow, etc. It is believed that all of these intrinsic image properties must be derived simultaneously (*i.e.*, the problems are tightly coupled) and that only massively parallel architectures such as the human brain are adequate for the task. These intrinsic images have been labeled "parameter networks" [Ballard81b], and the associated "constraint transforms" are modeled as Hough transforms. New methods of image analysis may arise from this work.

2.4. Related Applications

This section describes applications that are similar to Hough transform applications, but that differ in some fundamental fashion. While the difficulties with applying the transform could be overcome with special hardware, other techniques would often be more appropriate.

Cueing is the initial detection of interesting objects in a scene. Sometimes object

recognition is included with cueing when only certain types of object are of interest. The GHOUGH program can be used to cue instances of a given shape, but it is not well adapted to recognizing classes of objects. At best it can maintain a separate accumulator for each of several shapes or subshapes (*e.g.*, corners, straight lines, or parallel segments). Cueing is better performed with real-time level slicing or pseudo-coloring, or with matched filters [Otazo79], statistical classifiers [Touchberry77], blob detectors [Klein77, Tisdale77, Milgram78a, Deal79, Tisdale79], corner detectors [Perkins73, Rosenfeld78, Hwang79], or unusual-pattern detectors [Haralick75, Winkler78].

Object identification is the recognition of a shape once it has been cued. The GHOUGH program is only capable of template matching [Stockman77], and would have to maintain a separate accumulator for each identifiable silhouette. In gray-scale imagery this may be less successful than correlation matching; in perspective scenes the more flexible techniques of boundary following [Lucey76, Martelli76, Nahi78] or edge linking [Nevatia76, Milgram78b, Narendra79, MacVicar-Whelan81, McQueen81] and shape analysis [Arcelli71, Freeman74, McKee76, Tanimoto77, Binford81] may be superior.

Image-to-image registration is the process of finding the warp coefficients needed to align one image with another. A GHOUGH approach would require that (possibly unidentified) shapes found in one image be located in the other image so that warp parameters could be computed. This fails on low-contrast or blurred imagery and is less robust than correlation methods using all the gray-scale or gradient information in the scene.

Stereo compilation is a particular type of image-to-image registration in which the two images are known to differ by a small change in camera position. The compilation uses the known perspective change to compute distances from the camera to elements in the image. This and other stereo matching problems are basically one-dimensional, and are better handled by correlation or special-purpose techniques [Gennery77, Baker80, Grimson80] than by a general shape-detection routine.

Registration and cueing are just particular applications of pattern recognition. A great many pattern recognition techniques have been developed for recognizing regularities or particular patterns in lists of data vectors. Any of these techniques could be substituted for the accumulator technique used by the GHOUGH program. Some of these techniques offer adaptive improvement as they process new images, others are guaranteed optimal for particular detection problems.

Ellipse detection can be used as a particular example. The GHOUGH program treats ellipse templates as general shapes, no different from any others. Each edge element must vote for many cells in the accumulator – all the cells occupying a cone in (x,y,radius) space. These votes could be reduced to a single vote if edge pairs were considered instead of individual edges [Laws77]. Computational savings are even greater if ellipses of arbitrary aspect ratio are to be detected in a five-dimensional parametric space [Tsuji77, Tsuji78]. The tradeoff in these cases is that the number of edge pairs is much larger than the number of edge elements, so that complex screening is needed to reduce the number of edge pairs actually used.

Another advantage of some pattern recognition techniques is that the detected object is identified by arbitrarily accurate coordinates. There is no quantization problem or corresponding computational space/time tradeoff as with the Hough techniques. Two-pass coarse/fine registration techniques are thus not required, although back projection and validation against the image may still be desirable.

Section 3

Description

This section presents the history of generalized Hough analysis methods and a detailed statement of the GHOUGH algorithm. The historical information is intended to clarify the major issues in recursive segmentation and to provide entry points into the literature.

3.1. History

The GHOUGH program detects instances of a given shape by combining many pieces of local evidence. The original Hough transform [Hough62, Rosenfeld69, Griffith70] was a method of finding lines through a pattern of points. Duda and Hart [Duda72] introduced the normal parameterization of lines instead of the slope-intercept form. Others [O'Gorman73, Perkins73, Shapiro74] used gradient direction at each point to improve the voting method.

Other related methods have been used to locate circles [Bastian71, Duda72, Kimme75], parabolas [Wechsler75], and ellipses [Tsuji77, Tsuji78]. The extension to other analytic shapes was obvious, although implementation questions remained [Shapiro75, Shapiro76, Shapiro78a, Shapiro78b, Shapiro79a, Zabele79]. A survey of this work has been published [Iannino78]. Hough transforms have also been applied to trajectories generated by dynamic processes [Shapiro78c, Yam81].

Merlin and Farber [Merlin75] extended the Hough technique to nonanalytic shapes by using a template to control the gathering of evidence. Their method mapped each (directionless) point in a binary image into an inverted trace of the template in the accumulator. The best instance of the sought shape was then indicated by the cell accumulating the most votes.

Ballard [Ballard81a] restructured this technique to make it flexible and efficient. The GHOUGH program grew out of this work [Sloan80]. It uses a compiled template to simplify addressing in the accumulator, and it starts with directed edge segments instead of image points. The directed edge segments constrain the possible orientations of a shape instance so that only a few cells need to be incremented.

3.2. Theory

There have been many formalizations of Hough transform theory, particularly for the application of analytic curve detection. See van Veen [van Veen81] for a particularly nice analysis of straight line detection; a related error analysis has been published by [Shapiro79a]. It has recently been recognized that Hough transforms are a special case of the Radon transform used in computerized tomography [Deans81]; this may provide important theoretical methods similar to those of Fourier analysis. Fourier theory may also be applicable since the Hough voting pattern is similar to an optical point-spread function [Brown82a].

Description

All Hough techniques consist of the following elements:

* A shape specification with N-dimensional parameter space P;
* An N-dimensional accumulator, A, representing P;
* A local element detector, E, applied to the image;
* A voting rule, V, mapping the information of E into A;
* A detection rule, D, specifying conditions on A that indicate instances of the shape.

The GHOUGH program makes specific choices for these elements that determine its range of application.

Two image regions have the same shape if the outline of one can be generated by a combination of translation, rotation, or uniform scaling of the other. Each such set of equivalent shapes is represented by a template of normalized size and arbitrary orientation. (The template is scaled to have unit radius relative to the reference point described below.) A particular shape instance is thus represented by a particular template and four parameters: horizontal and vertical position, rotation, and radius.

A template differs from a shape in that it has an associated reference point. The reference point is selected by the user when the template is first digitized. (It could be selected automatically, *e.g.*, as the center of boundary mass.) Usually the reference is an arbitrary point near the center of the shape, but it can correspond to any invariant location or feature. Although it need not be inside, computational problems will be minimized if all distances to the boundary are small.

A template is also composed of disconnected points instead of a continuous outline. The points are usually entered by tracing an image region, possibly with the aid of spline interpolation. They can also be generated mathematically since they are completely independent of any gray-level information in the image. The points must trace the shape clockwise, and the sequence is currently required to closed on itself. The density of sampling along the shape outline is arbitrary, but the smoothness of the representation may be critical. Boundary direction through a point is assumed to be parallel to the line through the point's two nearest neighbors. A small error in this direction can result in poor search performance because of magnification in extrapolating to the template's reference position.

The generalized Hough technique compiles the template into a data structure called an R-table. This is a sparse representation of the object-centered location and orientation of each template edge point, stored as a list of lists. The main list is accessed by the rotational orientation of an edge element relative to the horizontal axis. Associated with this orientation is another list of vectors representing x and y offsets from the edge element to the template reference point. Duplicate or multiple offset vectors may be present. Together all the edge orientations and offset vectors are sufficient to reconstruct the template.

In operation, an edge detector is applied to all pixel neighborhoods in an image window. At present a modified Sobel operator is used. A Sobel operator measures the gradient at each image point by the root-mean-square response to two convolution masks:

```
-1  0  1        -1 -2 -1
-2  0  2         0  0  0
-1  0  1         1  2  1
```

Description

The modified Sobel uses the sum-of-absolute-value combination of the two weighted sums. Experience has shown that the two methods are nearly equivalent, although the root-mean-square method is a little more constant in its response to rotated edges.

The response at each pixel is compared to a threshold supplied by the user; if it is above threshold, the detected edge is permitted to vote for likely shape centers. The proper threshold setting depends on the scene content, the target to be detected, and the edge detector itself. The modified Sobel detector typically produces a log normal distribution of edge gradient strengths.

The edge direction is computed from the arctangent of the responses to the two Sobel masks. A detected edge is then (conceptually) rotated by the minimum permitted rotation for valid shape instances. This orientation gives an index into the R-table to find the associated list of offset vectors. The accumulator cells corresponding to each offset vector and permitted radius (scale factor) are incremented. (A separate accumulator plane is used for each rotation and radius.) Then the edge orientation is further rotated by the specified increment and the indexing and voting procedure is repeated. This continues until the full range of permissible shape rotations has been processed.

Thus any given edge-element may vote for many possible shape centers. The number of accumulator cells incremented is proportional to the number of scale factors and rotational angles permitted in the search. The accumulator cell with the most votes indicates the most likely location, rotation, and radius of the shape.

Since there may be any number of shape instances present, including none, it is necessary to decide which cells correspond to genuine occurrences. The GHOUGH program uses two criteria: each reported shape instance must be a relative maximum within the four-dimensional accumulator, and must also be above some threshold number of votes. A relative maximum is simply a cell with at least as many votes as neighboring cells. The threshold is required to prevent isolated votes (*e.g.*, single votes surrounded by no others) from being reported. At present all cells meeting these criteria are reported as shape instances, with those having the most votes reported first. Any higher level analysis is left to the program receiving this list of "possible target hits."

Section 4

Implementation

This section documents the SRI Testbed implementation of GHOUGH. It is intended as a guide for system maintainers and for programmers making modifications to the GHOUGH system. The terms used in this section may be a little cryptic: they are either defined elsewhere in this report or come from the supporting operating systems.

The SRI Testbed uses the EUNICE operating system, which is a Berkeley UNIX[1] emulator for VAX computers using DEC's VMS operating system. EUNICE was developed at SRI to permit simultaneous access to UNIX and VMS software and system services, and to implement improvements to UNIX such as significantly faster image I/O. EUNICE is now a commercial product maintained by The Woolongong Group in Mountain View, California.

The GHOUGH package is currently compiled as a single file, *ghough.c*, containing a main program and several subroutines. The underlying algorithm is very little changed from the original University of Rochester implementation, but the command interpreter, help system, and supporting image access and display utilities are all new. These external routines are all part of the Image Understanding Testbed environment.

The main program and related files are in directory /iu/tb/src/ghough. Subdirectories *help* and *srchhelp* contain the text files used by the help system at the top command level and during the search command. Directory *template* contains predefined template files, and *demo* contains some simple shell scripts and "ground truth" files used in the evaluation effort.

Source code and help files for the CI driver are in /iu/tb/lib/cilib. For extensive documentation type "man ci" or run "vtroff -man /iu/tb/man/man3/ci.3c". The CI driver uses command-line parsing routines in cilib/cmuarglib and in /iu/tb/lib/sublib/asklib; both of these may someday be replaced by the Testbed argument parsing routines in sublib/arglib.

Other utility routines contributed by CMU have been distributed to /iu/tb/lib/dsplib/gmrlib, /iu/tb/lib/imglib, and /iu/tb/lib/sublib, and are documented in /iu/tb/man/man3. Some of these have been modified or rewritten for the Testbed environment: the image access code, for instance, reads Testbed image headers as well as CMU image headers.

To compile the GHOUGH program, just connect to this directory and type "make". You may type "make -n" to see what will happen if you do this. Additional options are documented in the header of the makefile.

The program contains two edge-detection routines. The first is a modified 3x3 Sobel operator; the second is a generalized 3Nx3N Sobel operator, where NxN block averages are substituted for the single pixel values used in the 3x3 Sobel. Both routines are

[1]UNIX is a trademark of Bell Laboratories.

meant to be applied at arbitrary image positions, as opposed to being optimized for moving-window application to an entire image plane. The interactive driver allows either edge operator to be applied to any image point. (This may help in threshold selection.)

The original contribution used a 6x6 version of the Sobel edge detector in which each 2x2 subsquare was averaged to produce the 3x3 pattern used in an ordinary Sobel edge detector. This gave noise immunity and increased accuracy for large targets at the cost of much slower operation and of poor performance on small targets. The Testbed version has been recompiled to use the standard 3x3 operator.

The search algorithm itself is separated from the user interface routine that gathers the search arguments and checks them for errors. The search algorithm could thus be invoked by any other program that set up the proper environment. It takes an image and several arguments as input and returns an accumulator as output.

The accumulator analysis has also been separated into independent modules. First the accumulator is searched for local maxima. The list of maxima is sorted and then passed to a deeper level of the CI driver. The user can then apply any sequence of analysis and display commands to the sorted list.

The original contribution reported only the single shape instance receiving the most votes. SRI has modified the testbed version to report a list of matches ordered by the number of votes. This permits identification of multiple instances in one image, as when finding oil storage tanks in aerial imagery. It was also necessary to write routines for examining and saving the list of matches, and for comparing the list to a ground-truth database.

GHOUGH demonstrations have been set up in subdirectories *lake*, *circle*, and *ellipse* of /iu/testbed/demo. Just connect to the appropriate directory and run the *demo* command.

The *demo* program in the *lake* directory is fully automatic: just type 'demo' and it will proceed to find the lake. You may also run the *display* script to show the results of a previously run *demo* script. For more interaction, invoke the GHOUGH program directly (*e.g.*, by typing 'ghough') and issue the '< lake.cmd' command; you will then be asked to delimit a search area with the cursor and to provide other parameters and display commands.

The *circle* and *ellipse* directories are very similar, except that there is no predefined interactive version of the demonstrations. The ellipse finder takes a particularly long time to run, so you should normally just run the *display* sequence.

Section 5

Program Documentation

This section constitutes a user's guide to the GHOUGH package as it is implemented on the SRI Image Understanding Testbed. As with any reference manual, it has occasionally been necessary to refer to terms before they are defined and discussed in detail. A preliminary scan through the section may be helpful on the initial reading. Additional information is available on-line, as described below.

5.1. Interactive Usage

The GHOUGH algorithm is currently embedded in a subroutine to fill the accumulator with votes and another subroutine to analyze the vote pattern and report back likely shape instances. These routines are called by ghough.c, a driver package that allocates the accumulator array, sets up the required environment, and interacts with the user. The following is a sample session using the GHOUGH program.

```
ghough
```

This invokes the program. You need not specify the full directory path name for the executable file if the path is given in your standard ".cshrc" shell startup file. If you have no startup file, you may have to specify "/iu/tb/bin/ghough" or some other full path name.

```
Generalized Hough Transform

>
```

The system responds and waits for commands. GHOUGH uses a version of the CI driver, so that built-in features of that driver are available. Additional commands are provided for the GHOUGH image analysis capabilities.

```
> *

  draw                search
  erase               select
  open                sobel
  print               template
  quit                trace
                      help


> *=

  verbose         no
```

An '*' command lists all of the GHOUGH commands, and an '*=' command will list any user-accessible variables. Typing 'help' will give further information on the CI

command interpreter and the help subsystem, and 'help *' will give a list of all available help topics. Additional information may be obtained by typing help and the name of a topic. For example:

```
> help menu

  draw ---------- draw the template.
  erase --------- erase the overlays.
  open ---------- display the input picture.
  print --------- print the template table.
  quit ---------- clear the display and quit.
  search -------- find instances of the template shape.
  select -------- select one of the four overlay colors.
  sobel --------- find Sobel gradient at specified point.
  template ------ read a template file.
  trace --------- trace a shape to create a template.

  Use "verbose = yes" to enable additional printout
  during searches.


> help usage

  Commands issued in the following order should work.

    open ---------- display an image.
    template ------ read a template file.
    draw ---------- display the template shape.
    search -------- search for instances of the shape.
    quit ---------- clear the screen and terminate.
```

To open an image file and display it, type

```
> open

  Image file:  <bw.img>
```

The default name is bw.img, but any other name may be specified. The image name may also be given on the command line, as in "open /iu/tb/pic/plant/bw.img". The image may then be used for tracing a new template or for searching using an existing template.

To create a new template:

```
> trace

  Draw a closed shape (clockwise).
  Use <CR> to enter a point, q to quit.

  Specify the shape center, then <CR>.

    Reference point: [256, 256]
```

```
Enter points with <CR>, terminate with q:

  Point  0: [156, 256]
  ...

Number of points entered: 46.
Maximum radius = 63.49.

Output file name: <template.dat>
```

The x and y components of the distance between each edge point and the arbitrary center of the shape are stored in a template file. For help in using the cursor, type 'help cursor.'

Any number of points may be used to outline the shape -- the more points that define a shape, the more are available to vote for the shape. A large number of points in the R-table also causes GHOUGH to run slower. One is faced with a trade-off between computation time and precision in detection of the desired shape.

```
> template

  File name:  <template.dat>
```

An R-table is built from a stored template file. The default file name is template.dat, but you may specify any valid template file. This command is not needed if you have just entered a template by tracing an image object.

To draw a shape instance in the display overlay:

```
>  draw

  Center column:   (0 to 511) <364>
  Center row:      (0 to 511) <128>
  Template angle (degrees):  (-359 to 360) <0>
  Template radius (pixels):  (1 to 511)  <60>
```

The default position is in the lower-right quadrant of the screen, which is convenient if the image you have open is 256x256 or smaller. The template angle is a counter-clockwise rotation relative to the original orientation of the traced shape. Template radius is the radius of the largest circle centered on the reference point that just contains the shape.

To run the edge detector over the image and search for shape instances:

```
>  search (160,90),(210,170)
```

This command begins a search for the template shape in a window with the specified lower left and upper right corners. If the corner coordinates are not specified, they will be requested interactively; you may then use the cursor to indicate the processing window. The remaining arguments of the search command are given in interactive form below.

The search command builds an accumulator and fills it with votes for template instances. The accumulator may be regarded as a stack of two-dimensional arrays, one for each combination of acceptable template size and rotation. The stack of arrays is registered with the image window specified above. Shapes to be found need

not be entirely within this window, but the window must cover the shape centers.

```
Horizontal resolution:   (1 to 51) <1>
Vertical resolution:     (1 to 81) <1>
```

You are next asked for the spatial resolution of the search. Specify single-pixel resolution [the default] for the finest resolution currently available. Use a larger spacing in either direction for a coarse search that uses a smaller accumulator and requires less search time.

```
Minimum rotation (degrees):   (-359 to 360) <-3>
Maximum rotation (degrees):   (-3 to 360) <3>
Angular Resolution (degrees):  (1 to 7) <1>
```

Specify the acceptable range of template rotations and the angular resolution of the search. You may specify any whole number of degrees. Here we have allowed three degrees on either side of the original template orientation.

```
Minimum radius (pixels):   (1 to 999)  <60>
Maximum radius (pixels):   (60 to 999)  <66>
Radius resolution (pixels):  (1 to 7)  <1>
```

You must also specify the acceptable range of radii for instances of the template shape. This is a distance in pixels and not a magnification factor. The radius is that of the largest circle which just contains the shape. The program is expecting floating-point numbers and will quantize the template radius to any specified resolution.

```
Edge threshold:  (0 to 999) <120>
```

You must specify the weakest edge gradient that will be allowed to vote for potential template instances. You may specify 0 if you want to use the best edge fit at every image pixel, but this will slow the search and raise the noise level in the accumulator. Fewer edges are considered as the value of the threshold increases. You may want to use the Sobel command (before invoking search) to establish a practical value for the threshold.

```
Contrast (b/w/m):  <black>
```

Specify the contrast of the figure as black, white, or mixed. This argument determines how the edge gradient will be interpreted. If the figure appears dark on a light background then you should specify black; if it is light on a dark background then specify white. You may specify mixed to include either or neither of the conditions: each edge element will then vote for both a dark shape and a light one.

```
Examining the edge structure ...
```

The program is now searching through the window looking for edges above the specified strength threshold. Each edge that it finds is displayed on the monitor screen.

```
Searching for local maxima ...
```

The program is now searching the accumulator for local maxima. A list of these potential template instances will be made available for further analysis.

```
37 maxima found; highest count is 5.

Enter display commands:

>
```

The program has found 37 potential hits, and has created a new subcommand interpreter so that you may study the results before terminating the search routine. To see the available subcommands:

```
> help menu

  above --------- display matches above a given count.
  display ------- display a particular match.
  erase --------- erase the overlays.
  first --------- display the first n matches.
  level --------- display a particular level of matches.
  quit ---------- clear the display and quit.
  save ---------- save the overlay as an image file.
  score --------- tabulate "hits" in a ground-truth file.
  select -------- select one of the four overlay colors.
  top ----------- display the first n levels of matches.

  Use "symbol = yes" to turn on full-shape display, "no"
  to display only center points (except for the "display"
  command).
```

This 'help menu' command shows the names of commands available at this level. Enter '*' or '*=' for a more concise list, or 'help topic' for more information.

```
> save edges.ovl
```

If you want to save the edge overlay you should do it now. The current program cannot recreate this display if you go on to something else.

```
> first 4

 #    Column     Row     Angle    Radius   Count
 1      181      141        2      66.0       5
 2      181      143        0      61.0       4
 3      181      144       -2      64.0       4
 4      181      143        0      62.0       4
```

Here we have listed the first 4 local maxima. The count field shows the number of votes that each match received. (Note that the fourth is not necessarily better than the fifth.) The centers of these four matches are displayed on the screen.

```
> top 2

 #    Column     Row     Angle    Radius   Count
 1      181      141       2       66.0       5
 2      181      143       0       61.0       4
 3      181      144      -2       64.0       4
 4      181      143       0       62.0       4
 5      181      139       0       64.0       4
 6      181      144      -1       64.0       4
```

The top command displays a number of threshold levels in the accumulator. Here the top two levels are displayed. We could save this display with a save command.

```
>  level 1

 #    Column     Row     Angle    Radius   Count
 1      181      141       2       66.0       5
```

We can also display a single threshold level. 'Top 1' and 'level 1' give the same result, but the two commands differ for larger arguments.

```
> display 1

 #    Column     Row     Angle    Radius   Count
 1      181      141       2       66.0       5
```

Here we have displayed a particular match (the first). Both the shape center and the outline are displayed on the monitor screen.

You may also use the 'score' command to compare detected shape instances with a previously existing ground truth file. This file is currently just a list of all known instances of a particular target type giving position, rotation, and radius; for the lake in the plant image, this is just

```
181   143   2   64.2
```

The score command will search for this target among the first 40 detected shape instances. These commands are particularly useful for later evaluation of GHOUGH runs executed as background or batch jobs. For our sample session:

```
>      score lake.tru

 #    Column      Row      Angle     Radius   Count

        181        143        2        64.2   (known)
 1      181        141        2        66.0      5
 2      181        143        0        61.0      4
 3      181        144       -2        64.0      4
 4      181        143        0        62.0      4
 5      181        139        0        64.0      4
 6      181        144       -1        64.0      4
 7      182        142       -1        60.0      4
 8      182        142        0        60.0      4
 9      181        143        0        60.0      4

..      ...        ...       ..        ....      .

1 of the 1 known targets were found
in the best 37 Hough maxima.
```

This example shows the lake being reported as many matches, each differing only slightly in its combination of parameters. The GHOUGH program recognizes these matches as being equivalent, but does not average or otherwise combine them to get a more accurate determination.

The quit return terminates this driver level and gives control back to the search routine. The search routine offers one more service before returning control to the top-level command driver.

```
> quit

Output a Hough plane?  <no>
```

A plane of the accumulator array may be output as an image file for later display. This can be useful for debugging or for getting a feel for the nature of generalized Hough analysis.

```
> quit

Finished.
```

A quit command terminates the session. The monitor screen is cleared and the display processor is freed for other users.

5.2. Batch Execution

The GHOUGH program offers two methods of invoking prestored commands. The first is the invocation of CI command files. For example, you might give the command

```
> <lake.cmd
```

where the file lake.cmd contains the commands

```
open /iu/tb/pic/plant/bw.img
template /iu/rochester/template/lake2.dat
draw 384,128,-45,45.0
search ,,1,1,-3,3,1,60,66,1,120,black,no
```

In this case the GHOUGH program will open an image, read a predefined template, display the template, and search for matching instances. You will be asked for the search area since its coordinates were omitted from the search command. You will also be able to enter interactive commands to examine the search results and save the corresponding graphic displays.

The second method is to drive the entire GHOUGH session from an operating system script. A UNIX C-shell script might look like:

```
# Run the lake demo.
ghough <<!
  open /iu/tb/pic/plant/bw.img
  template /iu/rochester/template/lake2.dat
  draw 384,128,-45,45.0
  verbose = yes
  search (180,90),(210,170),1,1,-3,3,1,60,66,1,120,black,no
    save edge.ovl
    symbol = yes
    first 20
    top 1
    save top1.ovl
    top 4
    save top4.ovl
    display 1
    save match1.ovl
    score lake.tru
    quit
  quit
!
```

This script is designed to run without user interaction. It does display its output on the terminal and on the display screen, but does not wait for you to look at the data. (You may add 'push_level' commands wherever you want it to pause and wait for commands. A quit command will then return you to the script command file. Specifying 'quit 2' will terminate the entire run.)

The script example above contains save commands to save copies of selected graphic overlays for later viewing. To save the typed terminal output you should pipe the standard output to a file. The UNIX method for doing this is to add " >session.log" to the ghough command within the script or to the UNIX command line that invokes the script. You may also use the UNIX 'script' or 'tee' commands to route the typed output to a file and to your terminal.

The actual submission of this shell script is described in the UNIX Programmer's Manual. You should run it in foreground mode if you want to interact with the program. If you run it in background mode, be sure to pipe the output to a log file so that it won't appear on your terminal. On a UNIX system you can monitor the log file during execution (using a 'cat' or 'tail -f' command) to make sure everything is running smoothly, and you can halt the process or reconnect it to your terminal if you wish. At present there is no way to turn off the GHOUGH display commands, although this option is planned.

Section 6

Evaluation

This section documents the performance of the GHOUGH program in several hundred test runs on a variety of aerial imagery. We have extracted rules for using the current system and have evaluated its characteristics.

6.1. Parameter Settings

GHOUGH search options available to the user are presented below. Typical parameter values are specified and the effects of different values are explained.

Image Content:

The GHOUGH program is at its best finding large, well-defined silhouettes in otherwise unvarying imagery. Since this is not a very useful talent, it is well that the program is also able to find partially hidden objects in noisy or blurred images. Any type of imagery is suitable as long as a sufficient portion of each target boundary is detectable. The objects must usually be obvious to a human before GHOUGH will be able to find them reliably.

The quality of object detection can be partially controlled by setting a threshold (described below), but is ultimately limited by the edge detector used. The modified Sobel detector currently in use works very well on most imagery, but is not as sensitive as an adaptive or task-specific detector might be. It does not give subpixel resolution, and the gradient angle is not very reliable (especially for weak edges).

A particular requirement with this edge detector is that object edges be sharply defined. Objects with gradual, uncertain boundaries (*e.g.*, reservoir outlines or FLIR targets) or strong internal gradients (*e.g.*, domes or "hot spots") will usually be located, but the position and orientation reported may not match that chosen by a human. The reported shape may also depend on the edge threshold used.

The requirement of sharp edges does not imply that smooth, continuous object boundaries are required. The program is quite tolerant of noise in the outline and is able to find irregular, incomplete, or discontinuous shapes. The circle template, for instance, often responds to forest clearings, tree tops, road intersections, and curved embankments, as well as to square buildings and to image "hot spots." The irregularities in these image structures spread the vote cluster in the accumulator, but the local maximum may still be above the general noise level.

Shadow edges usually fit the requirement for strong, sharp edges. It is often easier to find a shadow than to find the object that cast it. This may be a useful cueing technique, but must be used carefully to avoid reporting objects at incorrect locations. A similar problem exists with high-resolution imagery: the position reported for a part of an object (*e.g.*, the circular top of a storage tank) may not correspond

to the position of the whole object.

These characteristics mean that the program is best suited for three tasks: locating industrial parts in high-contrast imagery; counting numerous, obvious, similar objects such as storage tanks, barracks, or microscopic particles; and precisely positioning a template when an approximate location is cued by the user or by another system. Even for these applications, the program must be supervised and its output edited. Other applications will require further development of the technique.

Template:

Template shape is dictated by the targets being sought. Although this is not under user control, it may be helpful to understand the effect of shape complexity on GHOUGH performance.

A template is first compiled into an R-table, and all matching is done against this data structure. The R-table is sorted by edge angle, with one or possibly more shape center offsets recorded for each angle. For a convex template these offsets form an orderly progression matching the order in which the points were traced. For an involuted template (*e.g.*, a star-shaped outline) they jump around as points from one part of the trace interleave with points from another part. The net result is that, during edge processing, the votes from a particular image edge also jump around as different rotations are considered. This increases paging activity and total execution time, but has very little effect on detection performance.

The performance of a template depends only weakly on the number of points it contains. The density of points along the template perimeter determines the accuracy with which template edge angles are known. This is because the angle at a particular template point is computed as the direction between its two neighbors. The density of points also determines the accuracy of accumulator votes since the template point used by an image edge element for its voting is the one with computed angle nearest to, but larger than, that of the image edge.

These two effects cause spreading of the accumulator votes for a given shape instance. For small targets (*e.g.*, seven pixels in radius) the spread is slight, but possibly significant because the number of votes is small. For large targets (*e.g.*, 64 pixels in radius) the spread is much greater but is balanced by the large number of voting edges. A template with as few as eight points can be used to find targets, but will find a large target only if one of the accumulator planes represents exactly the right rotation and radius and the density of detected edges is almost optimal. For more robust operation the number of template points should be at least half the target radius. Generally 32 points are quite sufficient.

Contrary to intuition, using more points in the template neither increases the execution time significantly nor improves performance. The only effect on execution time is that of indexing into the R-table to find the closest template edge: this is currently done by following a chain of pointers until the next larger angle is found, but the program could be written using a hash table or array structure for the R-table that would have a constant (and faster) accessing time.

The effect on performance is that more accurate template edge angles slightly improve vote clustering and rotational accuracy. Improved vote clustering is only important in borderline cases and might be better achieved by convolution smoothing of the accumulator prior to (or during) detection of the local maxima. Rotational

accuracy is more a function of target shape than of the accuracy of individual edge angles. Further, too dense a template will suffer from quantization effects as described below.

The most common method of creating a template is to trace a known target. The GHOUGH program has a cursor driver to make this possible. Unfortunately, the position of the cursor can only be read to the nearest pixel. For a sparse outline of a large target this makes no difference, but for a dense outline it means that a template point will usually differ from each of its neighbors by only a single vertical or horizontal pixel (or both). Thus the edge angle computed using its neighbors will usually be a multiple of 45 degrees. This introduces severe quantization errors that make it difficult to extrapolate the shape center from a given image edge element.

Another consequence of this quantization error is that the R-table will contain only a few angles, each having a list of many corresponding shape center offsets. This is in contrast to a normal template having many angular entries with only a few having multiple shape center offsets. The quantized template results in far more votes being cast, but the votes are very scattered and serve mainly to increase the accumulator noise level. (One test with a 256-point template produced cells with noise counts of 37 votes, compared with only 6 votes for a similar 32-point template.)

Four solutions to this quantization problem are obvious:

- Use fewer template points.
- Expand (zoom) the prototype before tracing it.
- Smooth the entered points using spline functions.
- Use each entered point as an approximation, and compute the true edge position and angle from the underlying image.

All of these methods could be used together. Only the first is currently supported by the GHOUGH software.

The quantization effect also occurs naturally for angular template shapes. A square, for instance, has four principal edge directions plus four minor directions at the corners. Each of the major directions is associated with numerous center offsets, and smoothing methods will not change this.

The quantization effect for an angular template increases the accumulator noise level. (The number of votes for a 10-point by 10-point square template is about nine times the number for a similar, smoothly curved one.) It also produces some surprising rotational effects due to angular rounding when indexing into the R-table. Suppose that a square template is used to find a square target against a uniform background. If both are aligned, the edges from the image will vote for points along the sides of the template square. If the image is tilted, say, 40 degrees to one side, the edges will vote for exactly the same template points and there will be little harm. If the image is tilted one degree in the other direction, however, the image edges will all vote for the corner points in the template. This will completely change the pattern of votes in the accumulator. The algorithm is robust enough that it may still report the correct position for the square, but the estimate of the rotation angle is likely to be incorrect.

This effect is noticeable when using a "corner" template to locate right angles in an urban scene. In one case the search orientation had been set for zero to 270 degrees in 90-degree increments. GHOUGH found a great many right angles, but reported some of them at 90 degrees off the true orientation. Further, any nearly vertical or

horizontal building edge could produce a spurious corner if some of the edge element directions were rounded up 90 degrees from others. This rounding effect will have to be corrected before GHOUGH can be used as a reliable corner finder.

Search window:

The user is asked to define a search window by specifying or pointing to a lower-left pixel and an upper-right pixel. The box connecting these defines an edge-detection area. The number of possible edge locations is thus

$$Window\ points = (columns)(rows)$$

The current 3x3 edge detector requires an additional one-pixel border of data; for this reason the edge-search window cannot be the entire image.

The user must also supply horizontal and vertical "search grains." These affect only the accumulator, not the edge-detection process. A search grain of 2 in each direction means that a single accumulator cell will be used for each nonoverlapping group of four pixels at unit search grain. This reduces the accumulator storage by a factor of four, and also reduces execution time, but limits the accuracy with which a shape instance may be found.

The number of positions at which a shape could be reported is

$$Template\ positions = \left(\frac{window\ size}{search\ grain} - 2\right)^2$$

for a square search window with side length divisible by the search grain. The 2 subtracted from the quantized window size represents a one-cell border around each accumulator plane. This border is used for indexing efficiency during searches through the accumulator. Future versions of the program may eliminate it or add it on instead of subtracting it from the specified window.

The general rule for setting the window size is to use the smallest possible window. It is usually faster and more accurate to do several small searches rather than one large one that includes targetless areas. A preliminary coarse search could be used to cue areas for more accurate searches.

A coarse search gives less positional accuracy than a fine one, but does not necessarily give poorer detection. Small errors in estimating edge directions (in either the template or the image) can produce large errors in estimating the shape center position. Often the smoothing due to spatial quantization overcomes the resulting cluster spreading. The search grain should be set to the largest value that will give acceptable resolution: 10% of the minimum target radius is suggested.

Orientations:

The user is also asked to specify the acceptable ranges for rotation and radius. Each is given as a minimum, maximum, and increment, where the increment behaves very much like the spatial search grain specified above. Again, specify the coarsest search that will give acceptable information. If targets differ greatly in size, it is advisable to do one search for small targets using a small radius increment and another for large targets using a large increment.

Evaluation

The total number of orientations permitted is

$$Orientations = (rotations)(radii)$$

Each orientation requires an accumulator plane with the number of template positions given above; thus

$$Search\ volume = (orientations)(template\ positions)$$

An additional one-cell border has been added to the search volume (rather than subtracted from it), so the total memory required is

$$Accumulator\ planes = (rotations + 2)(radii + 2)$$

$$Accumulator\ volume = (window\ points)(accumulator\ planes)$$

Detection performance in each plane is nearly independent of that in adjacent planes unless the rotation or radius increments are very small. Adding additional range or resolution in x, y, rotation, or radius does not degrade performance in the other dimensions. It does result in more potential "hits" being reported for the same number of targets because the Hough voting method often produces four-dimensional ridges or plateaus of identical counts. Any such cells that are not adjacent to higher cell counts will be reported as local maxima. Thus a finely quantized accumulator will often produce a pattern of false matches with lower counts around the true match cell. A fairly sophisticated four-dimensional analysis technique would be needed to suppress these false matches.

Template radius is specified as a floating-point number and may take fractional values. All other parameters are currently restricted to integer values. Requesting subpixel accuracy in the radius will occasionally improve detection performance, but the true target size will still be estimated poorly because target center position cannot be known to subpixel accuracy. The chief result of such a request is to increase accumulator size and execution time.

Threshold:

The GHOUGH program uses a modified Sobel gradient operator for edge detection: the root mean square of the horizontal and vertical gradients has been replaced with a sum of absolute values to make the operator faster. Gradients, or edge strengths, computed with this operator have an approximate log normal distribution with an image-dependent peak somewhere from zero to 1600. An image of cloudy sky may have a peak near ten; modal edge strength in a low-angle urban scene may be 200 (depending on contrast ratio).

The edge threshold should ideally be set just low enough to detect most target edges. Any additional edges detected in the scene only contribute to accumulator noise level, making it harder to detect true matches. In practice one cannot determine such a threshold level without having previously located the targets. A more useful rule is to set the threshold according to image type or previously measured edge density.

The number and density of edges detected in an image are sigmoid (s-shaped) functions of edge threshold similar to cumulative frequency histograms. GHOUGH operates best when 10% to 20% of the pixels are classified as edge points, although it will usually work well at any edge density above 6%. Some typical threshold values to

achieve specified edge densities are:

Scene Type	6%	12%	25%	50%
Cloudy sky	42	35	28	20
Aerial terrain	160	120	80	40
Aerial target area	200	180	120	60
Low-angle urban	260	200	140	90
Forest cover	280	220	160	100
Aerial urban	720	600	480	340

A more accurate determination may be made using the verbose option of GHOUGH to print out the edge density found at a given threshold. This test can be done on a small image area to choose a threshold for the entire image. (Future versions could use an adaptive threshold that varies with edge density in each portion of the image.)

Occasionally the position, rotation angle, or radius reported for a target will vary slightly as a function of edge threshold. This is because the spatial distribution of edges in an image may vary with the definition of what constitutes an edge. GHOUGH gives equal weight to all edge elements regardless of their relative strengths or continuity. This is both a strength and a weakness of the algorithm: it permits shapes to be found in very noisy imagery, but reports nonintuitive detections for smoothly varying intensity regions such as hot spots, specular reflections, rounded surfaces, and shorelines.

In general it is better to use too low a threshold: this will increase chances of finding target edges while only slightly increasing noise level, and the edges found are likely to be the most reliable ones. The main drawback is that low thresholds increase the time required to fill the accumulator with votes. A reasonable starting guess is a threshold of 120.

Contrast setting:

The user is also asked whether to search for dark (black) objects on light backgrounds, light (white) objects on dark backgrounds, or both (mixed). Specifying 'mixed' will increase execution time and slightly increase the accumulator noise level. Use the most restrictive specification that will do the job.

6.2. Performance Statistics

This section documents the performance of GHOUGH during Testbed evaluation trials. Formulas are given, where appropriate, but most of the information is subjective.

Edge processing time:

The processing time (in seconds) required to compute edge locations and increment the corresponding accumulator cells may be modeled as

Evaluation

$$\textit{Edge time} = .00036(\textit{window points}) + .0053(\textit{edges found})$$
$$+ .00019(\textit{accumulator entries}) + (\textit{additional paging time})$$

where

$$\textit{Edges found} = (\textit{window points})(\textit{edge density})$$

$$\textit{Accumulator entries} = (\textit{edges found})(\textit{entry density})$$

$$\textit{Entry density} = (\textit{accumulator planes})(\textit{contrasts})$$
$$(\textit{quantization factor})(\textit{border effect})$$

The quantization factor is unity for a circle or smoothly varying template (whether convex or involuted), and is approximately one eighth the number of template points for a square. The border effect, or percentage of votes falling within the accumulator, is more difficult to estimate. It is typically near unity, but may be less than .25 when a large target is sought in a small image window.

Window points, edges found, and the number of accumulator entries are all nearly linear predictors of each other, but the proportionality constants depend on edge density and entry density. The "edges found" coefficient is slightly dependent on template density and on rotation range because of the time required to locate the proper angle in the R-table. A fraction of the time for "accumulator entries" is due to instrumentation code inserted for the evaluation; the rest is largely spent computing cell addresses.

A "normal" amount of virtual memory accessing is included in the above formulas. For very large accumulators there will be substantial amounts of additional paging time. This time increases with accumulator volume, edges found, system load, and with various system parameters. It also increases with increasing template radius or rotation range because the votes for template centers "jump around" more in the accumulator. Involuted or angular template shapes may also cause paging for this reason.

An additional penalty due to this extra paging is not included in the above equation: "real time" spent waiting for page requests to be satisfied. If other jobs are running on the system, this time may not be wasted, but it will delay completion of the GHOUGH task. For very large accumulators (*e.g.*, two million cells), the real time on an unloaded system may be several times the processing time used.

Accumulator search time:

Let

$$\textit{Accumulator density} = \frac{\textit{accumulator entries}}{\textit{accumulator volume}}$$

be the average number of entries in an accumulator cell. For densities below 0.5 the time required to search the accumulator for local maxima is almost entirely that required to examine each cell once and to sort the resulting list of maxima. This is because few cells reach the threshold of three counts required for further processing. A good formula is

Evaluation

$$\textit{Analysis time} = .000042(\textit{search volume})+.0025(\textit{maxima found})$$

This time is typically small and nearly constant.

For accumulator densities above 0.5 the search time depends on the density. Two opposing tendencies may be at work: more cells with at least three counts are found, but, with increasing density, the average number of neighbors that must be checked decreases (since there is a greater chance that a given neighbor has a count greater than the original cell). The net result may be modeled as

$$\textit{Analysis time} = 10^{-4}(\textit{search volume})(.08+2.6\log(1+\textit{accumulator density}))$$

$$+(\textit{additional paging time})+.0025(\textit{maxima found})$$

where the logarithm is to the base 10. The additional paging time depends on accumulator size, compactness (*i.e.*, search volume/accumulator volume), system load, and other system parameters. It may exceed 60 seconds in some cases, but is usually much smaller than the paging time during edge processing.

Local maxima:

The number of local maxima in the accumulator search volume is dependent on the "noise statistics," or the distribution of false counts in the accumulator. For unit search grain and a convex template, the number of maxima having at least three counts is approximately

$$\textit{Maxima} = .023(\textit{search volume})^{0.8}(1+\textit{accumulator density})^2-1$$

This number ranges from none to more than 6000. (The reporting of multiple maxima was added to GHOUGH as part of the Testbed evaluation effort. Future versions may use an adaptive threshold to prevent such numbers of false "hits" from being reported.)

At coarser search grains, the number of maxima drops much faster than this formula brings it up. Increasing spatial coarseness in the accumulator increases the number of local maxima until the accumulator density is about 1.0, then decreases it until there is only one maximum (or none) in the accumulator. As a rough approximation, using a search grain of two may double or halve the number of maxima found with unit search grain; further halving of the resolution will drop the number of maxima by a factor of five.

An angular template will also produce accumulator densities high enough to invalidate this formula. Halving the search resolution will again cause the number of maxima to drop by a factor of five or more, with no special exemption for a search grain of two.

Noise threshold:

The highest cell count due to noise is related to the search volume since larger volumes mean more chances for coincidental clusters of votes. Scene content (other than edge density) has little effect on noise statistics. A reasonable model is

$$\textit{Noise} = 2.04(\textit{search volume})^{0.09}(1+\textit{accumulator density})^{0.8}-1$$

which is usually accurate to within one count for noise counts of ten or fewer. For

most purposes, the noise threshold is small and nearly constant, but use of coarse search grain can push it into the hundreds. This formula seems to work for angular templates, but oversampling of a circular template (producing severe quantization errors) can increase the noise level by 50%.

The number of "noise maxima" found at successive threshold levels tends to increase by a factor of five. Thus a single cell with ten votes will be accompanied by five cells with nine votes, 25 with eight votes, etc. This pattern is not particularly reliable, particularly at high accumulator densities, but it could be used as the basis for an adaptive threshold: increase the threshold and discard all lower local maxima whenever, say, 50 maxima have been found above the current threshold. A tighter limit could be used if the number of valid targets could be estimated.

Detection performance:

As yet there exists no quantitative model for the number of votes required to indicate a shape instance. Some of the factors controlling the maximum count due to a target match are: average accumulator noise count, template shape, quantization factor, density of template points compared to positional and orientational resolution, target radius, position of the target relative to the quantized spatial grid, width of the target border and gradient across it, and percentage of border edges detected.

The GHOUGH program does not currently offer a threshold for automatically screening good matches from bad ones. If one were installed, the proper setting would be just above (or perhaps just below) the computed noise threshold. Higher-level intelligence would still be needed to screen matches reported by GHOUGH.

GHOUGH match reliability has been estimated by searching for targets (circles, ellipses, and a lake outline) in various images that did, or did not, contain such targets. A target was considered to be "found" if GHOUGH reported one or more matches within 15% of the true size at a location close enough (again, 15% of the true radius) to the known position. Angular accuracy was not checked, nor was the cell count compared to any computed noise threshold.

In one series of tests, the program searched for twelve small, nearly circular ellipses. The aspect ratio, rotation, and radius were known, although some searches permitted a small range for each parameter. The search window was either 128x128 or 254x254, and the "known" positions of the twelve targets were defined within each of these search windows (even for images without targets at these positions). A true "hit" or a false match (depending on the image) was reported if the "known" targets were among the first 20 local maxima.

False matches in images not containing targets were rare: no more than one of the "known" targets was ever reported, and these false matches had vote counts within the noise level. The false matches occurred in a view of urban San Francisco with the edge detector threshold set for very high edge densities (.45 to 1.00).

True matches and missed targets can only be reported for the scene containing the targets. For most runs this was the *plant* picture provided with the original GHOUGH program. It was very difficult to locate all twelve elliptical storage tanks in the plant image, probably because the edges are blurred and the radius is only 7 pixels. Detection of all twelve required a search grain of 2 as well as a critical edge detector threshold. At unit search grain, eleven of the twelve could be found at edge threshold 120, and eight or more were found with any threshold of zero to 180 (edge density 1.00 to

.10).

The above tests involved searching for dark objects on light backgrounds. Setting the contrast parameter to 'mixed' doubled the accumulator vote density, with a corresponding increase of a few counts in the noise level (*e.g.*, from six to seven). This did not prevent finding eleven of the twelve storage tanks (at the optimum edge density) since they had as many as 14 votes. The more general search also turned up a few white objects against dark backgrounds that were quite reasonable "hits."

Circles were also sought in various aerial images containing cylindrical storage tanks. Several images of Fort Belvoir showed multiple white storage tanks ranging from two to 19 pixels in radius. These tanks were extremely obvious to a human observer, but the GHOUGH program had great difficulty in finding them. Various parameter combinations were tried, but some targets resisted detection even when the search was tightly constrained to exactly the right location and size. Further, it was impossible to predict from visual appearances which of the tanks would be difficult to find; neither size nor contrast against background was a good criterion.

For example: the *ftb1a* image contains 29 clearly visible storage tanks in three clusters. The circular (or dome) tank tops range from two to six pixels in radius. A file was constructed giving the true location and radius of the tank tops as presented in this image. Attempts to find all of the tanks in one run proved futile. Some tanks were missed in each cluster, and one cluster was particularly difficult to match.

The obvious next step was to search through each cluster independently. This did improve detection probabilities, but results were still disappointing. One representative search over a cluster of 14 tanks located only three of the targets (to within 15% of true radius) in the first 40 matches. These matches were accompanied by numerous "almost" matches and a few wild "hits" that were difficult to perceive even after they were pointed out.

The *ftb1b* image is an enlargement of this cluster of 14 storage tanks. The circular tops range from eight to 19 pixels in radius. A search similar to that above was done using unit search grain and unit resolution in radius to find white circular objects. This time the "score" function reported multiple matches on six of the 14 tanks. Another three tanks were almost matched in the first 40 local maxima, but either the ground truth file was slightly inaccurate or the 15% scoring tolerance was too tight for the score function to report them. The remaining five tanks were matched very badly or not at all.

There is no reason to believe that software errors were responsible for the poor performance. A reasonable explanation involves the cylindrical nature of the targets: since they were viewed off-axis, they presented a roughly elliptical aspect consisting of the circular top overlapping the circular base. The accumulator thus gathered conflicting evidence for circles matching the top, the base, and various approximations to the elliptical composite, all with quantization errors due to the full-pixel spatial resolution. The spatial and orientational search grains were coarse enough that only one of these shapes could be reported, and very often the "wrong" shape had the most support.

Two improvements to the program might have prevented this situation: allow subpixel spatial resolution and analyze each accumulator plane independently so that existence of one circle does not necessarily imply nonexistence of similar circles. These and other suggestions are elaborated below. For the current implementation, however, the lesson is that the set of targets reported by the program must be

screened and edited by a more intelligent system. GHOUGH may assist a photointerpreter, but cannot yet replace him.

Section 7

Suggested Improvements

The process of evaluation has turned up numerous ways to improve the current GHOUGH implementation. Comments about existing features have been made at the appropriate points throughout this document. The following are additional suggestions for substantial modifications or needed research. Some of these would require major research projects or are beyond the scope of the original program. (The large number of suggestions should not be taken as a criticism of the GHOUGH system. Rather it is a tribute that the approach is flexible enough to support such extensions and is promising enough to be worth the effort.)

- *Linear Templates*

  Evaluation of GHOUGH as a corner detector was done using a right-triangle template having 20 points along each for the shorter sides and no points along the hypotenuse. This prevented the diagonal side from having a significant effect on the analysis, but a better solution should be implemented. The current restriction to closed templates is unnecessary. A simple modification would permit line segments, corners, riverbanks, and other fragments to be located. Multiregion templates (*e.g.*, outlines of machine parts with holes) should also be permitted.

- *Composite Templates*

  The current GHOUGH program searches for only one simple shape at a time. The principal author [Ballard81a] has suggested ways of searching for composite shapes, such as vehicles with visible wheels. One method uses the union or difference of R-tables to search for a particular composite. Another uses subtemplates independently to increment a common accumulator and then combines the evidence by convolving the accumulator with a special composite mask. These methods are nearly optimal for finding predictable, unarticulated shapes in low-quality imagery.

- *Subtemplate Accumulators*

  In higher quality imagery it may be better to search for the subtemplates independently using different accumulators and then invoke higher-level logic to combine the evidence. This permits sophisticated scene analysis, but the large number of generalized Hough analyses would consume both memory and computer time. Specialized Hough line, circle, and corner detectors should therefore be used.

Suggested Improvements

- *Improved Template Entry*

  The current method of entering templates using a keyboard-controlled cursor is clumsy. The program should be interfaced to a digitizing tablet or other tracing device. Tracing should be integrated with zoom capability to increase accuracy and reduce quantization errors. There should also be retrace and interpolation commands to simplify improvement of existing coarse templates. Entry of boundary points with subpixel accuracy and reliable edge angles should be supported.

- *Automated Template Extraction*

  Search performance could be improved by extracting template edge element directions from the image during initial tracing. The user would indicate an approximate boundary point, and the system would identify the nearest good boundary point and the gradient direction at that point. The increased accuracy of the edge positions and the edge directions at those points would reduce the number of boundary points needed to obtain a given search accuracy. This, in turn, would reduce execution time. A related capability, particularly useful for tracking applications, would allow a new template to be extracted automatically after GHOUGH had registered an existing template with the image.

- *Gradient Map Input*

  The current implementation of edge detection is too slow. The program selects a pixel position, reads in the neighborhood data, computes the edge gradient, and, possibly, increments the accumulator. It then moves to the next pixel and begins again with an overlapping neighborhood. This is inefficient, particularly with neighborhood windows larger than 3x3. For most applications it would be more efficient to use a separate moving-window gradient operator to provide gradient or edge map input to the GHOUGH program. The same map could then be used to locate many different objects using separate runs of GHOUGH or other programs.

- *Adaptive Edge Threshold*

  The user must set a threshold value for choosing which gradients are strong enough to be useful. Too low a threshold will introduce noise and slow the computation; too high a threshold will miss low-contrast edges. Future versions of the program should use adaptive thresholds instead of this constant global threshold.

- *Adaptive Edge Resolution*

  The current edge detector might also be improved. The 3x3 Sobel edge detector is excellent for finding small objects with sharp edges, but is nonoptimal for large objects and gradual edges. Use of different detector sizes requires separate GHOUGH programs to be run, and there is no way to combine evidence from multiple runs. The best solution would be a hierarchy of

edge detectors of different sizes all applied to each point in the image. The edges found at different resolutions could each be allowed to vote for shapes, or the single edge with the strongest support could be allowed to vote. It might also be possible to incorporate line detectors, corner detectors, and edge curvature measures for even better performance.

- *Confidence Weighting*

Each detected edge element votes once for each shape that could contain it, and each edge element has equal importance regardless of the strength of the edge. It may be desirable, as a user-selected option, to assign more weight to the edge elements that are more certain [O'Gorman73, van Veen81].

- *Negative Voting*

The problem of finding local maxima in the accumulator is necessitated by the use of positive weights for all votes. If the voting pattern were zero-mean, incorrect (or noise) cells would tend toward zero counts while true peaks would continue to accumulate positive votes. This technique, called CHOUGH or Complementary Hough, has recently been investigated by Brown at the Univ. of Rochester [Brown82a, Brown82b]. He recommends that each positive vote be accompanied by negative votes of 1/2 on each side across the direction of the voting outline. This represents the fact that positive evidence for one shape is also negative evidence for other similar shapes. Brown has also developed an analogy between GHOUGH voting patterns and standard analytic techniques in optics. (CHOUGH voting patterns are similar to the diffraction patterns of coherent optics, and also to the lateral inhibition patterns present in neurological vision systems.)

- *Sparse Edge Detector Application*

Coarse search is currently implemented by combining cells in the accumulator. This greatly reduces computer memory requirements but only slightly reduces execution time. An additional saving could be achieved by applying the edge operator at every nth row and column to match the spatial grain of the accumulator. This saving would be small if the edge operator were still applied at full resolution.

- *Multiple Resolution Analysis*

An even simpler approach to coarse search would use a "pyramid" of images at different resolutions. This would eliminate the need for a search grain increment in applying edge operators and indexing the accumulator. The program would start with a highly reduced image (or with an edge map of such an overview image) and would attempt to find any objects visible at that resolution. It would then use a local search in a higher resolution image to confirm and precisely locate the objects. (A disadvantage of any coarse search method is that, once an object is missed, it may never be found at higher resolutions.)

Suggested Improvements

- *Arbitrary Windows*

  The current restriction to rectangular search areas is due to the implementation of the accumulator as an array registered with the image window. There are other implementations, perhaps more expensive, which would allow edge-search windows of arbitrary shape. At the least it should be possible to specify multiple rectangular windows instead of having to run the GHOUGH search separately for each area or jointly with a much larger accumulator.

- *Independent Search Areas*

  In the original GHOUGH implementation, the user was asked to specify a shape-center search area within the edge-search window. This helped maintain resolution when estimating the position of a large object in a slightly larger edge-search window. If this option is reinstated, the center search window should not be restricted to be within the edge search window, or even within the image.

- *Arbitrary Resolution*

  The spatial resolution is currently limited to whole pixels, and the angular resolution is limited to whole degrees. The radius scale factor, on the other hand, may be specified to any desired resolution. There is no inherent reason why all four parameters should not be kept in floating-point, thus permitting very fine shape matching when the approximate match position is already known. Edge detectors with subpixel accuracy (*e.g.*, [MacVicar-Whelan81]) could also be used for fine searches.

- *Storage Reduction*

  Greater spatial and orientational accuracy are possible if the accumulator uses fewer bits per cell. The accumulator now consists of 32-bit integers, but should be reduced to 16-bit integers. The highest count encountered in numerous runs was 678, obtained with a square template applied to an urban scene with a search grain of 16 in each direction. At unit search grain, the highest counts rarely exceed 40, or even 20. A hardware implementation might manage with five bits per accumulator cell, or perhaps with even fewer by using logarithmic counts with stochastic updating [Morris78].

- *Logarithmic Radius Spacing*

  The specification of radius range by a minimum, maximum, and increment makes it very difficult to maintain resolution while covering a useful range. The spacing between sequential radius values should be logarithmic instead of incremental. With certain accumulator analysis schemes, the search grain could also be made proportional to radius.

Suggested Improvements

* *Adaptive Accumulator*

  Another method for increasing the resolution is to use an adaptive accumulator [O'Rourke81a, Sloan81]. Such an accumulator is continually restructured as the votes are entered so that high resolution is maintained in those areas that are receiving many votes. This would permit precise matching in a single pass instead of using a coarse search followed by a fine search. The cost in algorithm complexity might be quite high, however.

* *Hash-Table Accumulator*

  An attractive alternative to a full adaptive accumulator is one based on a hash algorithm. Only those cells that have significant counts are maintained; others are flushed periodically or as accumulator space is needed. Although the best accumulator size and flushing strategies are not yet known, Brown reports encouraging results [Brown82c].

* *Accurate R-Table Indexing*

  The current algorithm for using the R-table always rounds angles up to the next larger table entry. This is acceptable for many purposes, but disastrous with an angular template such as a square or right angle. The algorithm should be modified to round to the nearest table entry. This improvement could be combined with a change in the R-table format; a hash table or ragged array representation could be referenced faster than the current linked-list format.

* *Efficient Accumulator Indexing*

  The accumulator is currently accessed by column, row, rotation, and radius, with the radius subscript varying most quickly. A more efficient subscripting order might be found, although this one seems reasonable. A sizable improvement in paging performance might be possible using Quam's method of four-dimensional block storage [Quam80]. This would reduce the number of accumulator entry page faults caused by cycling the rotation and radius associated with each image edge element. The efficiency gained would depend on the number of storage words in each virtual memory page.

* *Elimination of Accumulator Padding*

  The accumulator search should be modified so that points outside the search volume are never examined. This will increase the accumulator search time and search complexity, but the effect will be small. In return the accumulator volume and the number of entries into the accumulator will be greatly reduced, sometimes by a factor of nine. This will also reduce system paging time, which can be a major component of the total run time.

Suggested Improvements

* *Accumulator Save/Restore*

GHOUGH allows individual planes of the accumulator to be written out as images. This capability should be augmented so that the entire accumulator could be written out and later restored. Researchers could then test various accumulator analysis algorithms without having to repeat the edge detection and accumulator building steps. There should be a similar capability for saving and restoring the list of matches. These modifications suggest a slightly different user interface than currently exists; one where the setting or modification of search parameters is a separate step from the activation of routines using those parameters.

* *Adaptive Match Screening*

Too many potential matches (*e.g.*, over 6000) are often reported, partly because of lack of smoothing (see below). The program should discard poor matches when much better ones are found. There seems to be no reason for reporting more than a few hundred, or even a few dozen, potential matches. The required screening could be made automatic, or could be controlled by user-settable parameters.

* *Accumulator Smoothing*

Votes for a single instance of a large shape tend to be reported as multiple maxima (when using the 3x3 edge detector). Both the spreading of the true peak in the accumulator and the reporting of false maxima due to noise can be combatted by blurring the four-dimensional accumulator values before attempting to find local maxima. The program authors recognize the need for such blurring [Sloan80, Ballard81a], but do not suggest any particular convolution function. A simple center-weighted 3x3x3x3 mask may be sufficient.

* *Relaxation Enhancement*

The smoothing suggested above may exacerbate the problem of one shape instance suppressing the detection of other similar instances (*e.g.*, overlapping circles in images of storage tanks). Either simpler or more complex relaxation and reinforcement processes might be more effective for particular applications.

* *Cluster Analysis*

Another possibility would be to search for maxima independently in each accumulator plane and then combine the resulting lists. This might offer great savings in accumulator analysis time if the planes used were aligned with the data storage order. The current "score" routine recognizes matches as being equivalent if they are within tolerance of a known target. A more sophisticated algorithm could determine clusters of reported matches without knowing ground truth.

Suggested Improvements

- *Match Quality Measures*

  At present the only information reported about a match is the number of votes it received. The reporting algorithm could also determine peak sharpness, four-dimensional volume, parameters of the best fitting paraboloid, peak-to-sidelobe ratio, or other descriptive statistics. These could be used by a higher-level program to screen or combine the reported matches and to estimate the true position in the accumulator-to-subcell accuracy. The quality of a match might also be estimated from the cell count since each cell has a theoretical maximum count that is dependent on the noise level and the completeness of the corresponding shape instance.

- *Match Verification*

  Future versions of the GHOUGH program may also project matches back into the image to check for continuity [Kitchen81] or to use a warp-tolerant match verification such as chamfer matching [Barrow77] or elastic matching [Burr81]. Higher-level knowledge such as the plausibility of occluding objects [Chien74] or of particular target configurations [Fischler73, Brown79, Price81] might be used for validation.

- *Multiframe Validation*

  Another possibility is multiframe validation [Iler79, McIngvale80]. A reliable match will be consistently reported, whereas an unreliable match will be reported at different locations in different frames. The frame-to-frame statistics of a match may thus be used to determine match quality. The number of frames that can be used depends on the rate at which frames are matched, the speed with which registration positions normally change, and the available time for target lock-on.

- *Fuzzy Display*

  GHOUGH displays each detected match as a sharp shape instance corresponding to the accumulator cell indices of the local maxima. The spatial and orientational quantization of the accumulator makes this a misleading representation. The shape could be displayed as a somewhat fuzzy overlay indicating the ambiguity in true target location. A better solution, of course, would be to locate the target more precisely either by interpolation in the accumulator or by match verification in the image.

- *Improved Score Function*

  The command that compares detected matches to expected target locations should similarly be modified to take search resolution into account. It makes no sense to reject a match with a four-pixel error if the search grain is eight pixels.

Suggested Improvements

- *Improved User Interface*

  The GHOUGH program allows the user to examine the list of detected targets using several screening criteria. Two extensions are needed: (1) storage and later recovery of the list for delayed analysis and (2) additional screening routines based on radius, orientation, match quality, or arbitrary combinations of factors. A separate command language or editing system should be developed for working with these lists. A LISP language might make a good basis.

- *Artificial Intelligence*

  Feature-matching systems have always been limited by the quality of the feature-extraction process. Future systems may integrate feature extraction with feature matching so that high-level considerations can direct the application of low-level image operators [Perkins73]. The high-level analysis may even direct the gathering of additional imagery. This falls within the planning and expert system areas of artificial intelligence [Garvey74, Garvey76, Brown79].

Section 8

Conclusions

The GHOUGH program is an efficient and flexible template-matching system. Its greatest potential is in cueing, counting, and mensuration applications. Its weaknesses are mainly correctable, although the inherent quantization of the Hough accumulator space does limit accuracy on any single pass through an image.

The current implementation requires a large number of user-supplied parameters in order to limit the search space. This is inconvenient for simple interactive use, but necessary given current computer resources. The ability to focus on specific recognition problems will be of more use if generalized Hough techniques are built into sophisticated or intelligent systems.

We have documented the GHOUGH program, evaluated its performance, and suggested ways to improve the algorithm and its implementation. Analytic models have been provided where appropriate. While the existing system has definite limitations, it demonstrates many promising uses for the GHOUGH algorithm.

Implementation of the GHOUGH program on the DARPA IU Testbed required extensive modification of the user interface and had considerable influence on the Testbed itself. The merging of the GHOUGH code from the University of Rochester with graphics subroutines and the CI command interpreter from Carnegie-Mellon University was a validation of the Testbed concept.

References

[Arcelli71] C. Arcelli and S. Levialdi, "Picture Processing and Overlapping Blobs," *IEEE Trans. on Computers*, pp. 1111-1115, Sep. 1971.

[Baker80] H.H. Baker, "Edge Based Stereo Correlation," *Proc. Image Understanding Workshop*, College Park, MD, pp. 168-175, Apr. 1980.

[Ballard76] D.H. Ballard and J. Sklansky, "A Ladder-Structured Decision Tree for Recognizing Tumors in Chest Radiographs," *IEEE Trans. Comput.*, Vol. C25, pp. 503-513, 1976.

[Ballard81a] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, Vol. 13, No. 2, pp. 111-122, 1981.

[Ballard81b] D.H. Ballard, "Parameter Networks: Towards a Theory of Low-Level Vision," *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 1068-1078, 1981.

[Ballard81c] D.H. Ballard and D. Sabbah, "On Shapes", *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 607-612, 1981.

[Barrow77] H.G Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," *Proc. Image Understanding Workshop*, pp. 21-27, Apr. 1977.

[Bastian71] P. Bastian and L. Dunn, "Global Transformations in Pattern Recognition of Bubble Chamber Photographs," *IEEE Trans. on Computers*, Vol. C-20, p. 995, Sep. 1971.

[Belsher79] J.F. Belsher, H.F. Williams, and R.H. Kin, "Scene Matching with Feature Detection," *Digital Processing of Aerial Images*, SPIE Vol. 186, pp. 12-20, 1979.

[Binford81] T.O. Binford, "Spatial Understanding," *Proc. Image Understanding Workshop*, Washington, D.C., pp. 236-240, Apr. 1981.

[Bolles81] R.C. Bolles and M.A. Fischler, "A RANSAC-Based Approach to Model Fitting and its Application to Finding Cylinders in Range Data," *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vancouver, pp. 637-643, 1981.

[Bolles82] R.C. Bolles and R.A. Cain, *Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method*, SRI Int., Tech. Note 262, Mar. 1982.

[Brown79] C.M. Brown, K.A. Lantz, and D.H. Ballard, "Model Driven Vision using Procedure Description," *U. of Rochester Computer Sci. Engineering Review*, pp. 16-24, 1978-79.

[Brown82a] C.M. Brown, *Bias and Noise in the Hough Transform I: Theory*, TR 105, Computer Science Dept., Univ. of Rochester, June 1982. A summary has been submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence*. as *Inherent Bias and Noise in the Hough Transform*.

[Brown82b] C.M. Brown, *Bias and Noise in the Hough Transform II: Experiments*, TR 113, Computer Science Dept., Univ. of Rochester, Aug. 1982.

[Brown82c] C.M. Brown and D.B. Sher, "Modeling the Sequential Behavior of Hough Transform Schemes," *Proc. Image Understanding Workshop*, Stanford, pp. 115-123, Sep. 1982.

[Burr81] D.J. Burr, "Elastic Matching of Line Drawings," *IEEE Trans. on Pattern Anal. and Machine Intelligence*, Vol. PAMI-3, No. 6, pp. 708-713, Nov. 1981.

[Chien74] R.T. Chien and Y.H. Chang, "Recognition of Curved Objects and Object Assemblies," *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition*, Copenhagen, pp. 496-510, Aug. 1974.

[Chien75] Y.P. Chien, "On the Optimal Extraction of Boundary Curves," *IEEE Conf. on Computer Graphics, Pattern Recognition, and Data Structure*, Los Angeles, pp. 208-209, May 1975.

[Choate79] W.C. Choate and W.W. Boyd, "Correlation Tracking Concepts for THASSID," *Proc. IEEE 1979 Nat. Aerosp. and Electron. Conf. (NAECON 1979)*, Dayton, OH, pp. 79-85, May 1979.

[Clark79] C.S. Clark, W.O. Eckhardt, C.A. McNary, R. Nevatia, K.E. Olin, and E.M. VanOrden, "High-Accuracy Model Matching for Scenes Containing Man-Made Structures," *Digital Processing of Aerial Images*, SPIE Vol. 186, pp. 54-62, 1979.

[Davis82] L.S. Davis, "Hierarchical Generalized Hough Transforms and Line-Segment Based Generalized Hough Transforms," *Pattern Recognition*, Vol. 15, No. 4, pp. 277-285, 1982.

[Deal79] B. Deal, C.M. Lo, R. Taylor, V. Norwood, H. Henning, T. Daggett, T. Noda, J. Powers, G. Guzman, H. Greenberger, G. Towner, and G. Parker, *Automatic Target Cuer First Quarter Report*, Northrop Corp., Electro-Mechanical Division, Anaheim, CA, Report NORT-79Y100, 101 pp., Oct. 1979.

[Deans81] S.R. Deans, "Hough Transform from the Radon Transform," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 2, pp. 185-188, Mar. 1981.

[Doudani77] S.A. Dudani and A.L. Luk, "Locating Straight-Line Edge Segments on Outdoor Scenes," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, pp. 367-377, 1977.

[Dougherty79] L.S. Dougherty, "A New Technique for Tracking Sequences of Digitized Images," *Proc. IEEE Conf. on Decision and Control*, Fort Lauderdale, FL, pp. 1028-31, December 1979.

[Duda72] R.O. Duda and P.E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," *CACM*, Vol. 15, No. 1, pp. 11-15, 1972.

[Fischler73] M. Fischler and R.A. Elschlager, "Representation and Matching of Pictorial Structures," *IEEE Trans. on Computers*, Vol. C-22, No. 1, pp. 67-92, Jan. 1973.

[Fitts79] J.M. Fitts, "Precision Correlation Tracking via Optimal Weighting Functions," *Proc. IEEE Conf. on Decision and Control*, Fort Lauderdale, FL, pp. 280-283, December 1979.

[Flachs78] G.M. Flachs, P.L Perez, R.B. Rogers, S.J. Szymanski, J.M. Taylor, and Y.H. U, *Real-Time Video Tracking Concepts*, New Mexico State University, Las Cruces, NM, Report NMSU-TR-78-1, May 1978, 69 pp.

[Flachs79] G.M. Flachs, P.L Perez, R.B. Rogers, S.J. Szymanski, J.M. Taylor, and Y.H. U, *A Real-Time Video Tracking System*, New Mexico State University, Las Cruces, NM, January 1979, 180 pp.

[Freeman74] H. Freeman, "Computer Processing of Line-Drawing Images," *Computing Surveys*, Vol. 6, No. 1, pp. 57-97, Mar. 1974.

[Garvey74] T.D. Garvey and J.M. Tenenbaum, "On the Automatic Generation of Programs for Locating Objects in Office Scenes," *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition*, Copenhagen, pp. 162-168, Aug. 1974.

[Garvey76] T.D. Garvey, "An Experiment with a System for Locating Objects in Multisensory Images," *Proc. 3rd. Int. Jnt. Conf. on Pattern Recognition*, Coronado, CA, pp. 567-575, Nov. 1976.

[Gennery77] D.B. Gennery, "A Stereo Vision System," *Proc. Image Understanding Workshop*, Palo Alto, pp. 31-46, Oct. 1977.

[Goodman79] L.R. Goodman, "A General Model for the Multiple Target Correlation and Tracking Problem," *Proc. 18th IEEE Conf. on Decision and Control*, Fort Lauderdale, FL, pp. 383-8, December 1979.

[Grimson80] W.E.L. Grimson, "Aspects of a Computational Theory of Human Stereo Vision," *Proc. Image Understanding Workshop*, College Park, MD, pp. 128-149, Apr. 1980.

[Griffith70] A.K. Griffith, *Computer Recognition of Prismatic Solids*. Ph.D. Thesis, Dept. of Math., MIT, June, 1970.

[Haralick75] R.M. Haralick, "A Resolution Preserving Textural Transform for Images," *Proc. Computer Graphics, Pattern Recognition, and Data Structure*, Los Angeles, pp. 51-61, May 1975.

[Hemami70] H. Hemami, R.B. McGhee, and S.R. Gardener, "Towards a Generalized Template Matching Algorithm for Pictorial Pattern Recognition," *Proc. IEEE Symposium on Adaptive Processes (9th) Decision and Control*, Austin, TX, 4 pp., Dec. 1970.

[Hinton81] G.E. Hinton, "A Parallel Computation that Assigns Canonical Object-Based Frames of Reference," *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 683-685, 1981.

[Hough62] P.V.C. Hough, *Method and Means for Recognizing Complex Patterns*, U.S. Patent 3069654, 1962.

[Hwang79] J.J. Hwang, C.C. Lee, and E.L. Hall, "Locating Vertices in a Scene," *Proc. IEEE SOUTHEASTCON*, pp. 357-362, 1979.

[Iannino78] A. Iannino and S.D. Shapiro, "A Survey of the Hough Transform and Its Extensions for Curve Detection," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Chicago, pp. 32-38, May 1978.

[Iler79] T.W. Iler, M. Pitruzzello, and P.H. McIngvale, "Design and Evaluation of an Automatic Hand-Off Correlator," *Wkshp. on Imaging Trackers and Auton. Acq. App. for Missile Guidance*, Redstone Arsenal, AL, pp. 310-24, November 1979.

[Kimme75] C. Kimme, D.H. Ballard, and J. Sklansky, "Finding Circles by an Array of Accumulators," *CACM*, Vol. 18, pp. 120-122, 1975.

[Kitchen81] L. Kitchen and A. Rosenfeld, "Edge Evaluation using Local Edge Coherence," *Proc. Image Understanding Workshop*, Washington, D.C., pp. 99-113, Apr. 1981.

[Klein77] G.M. Klein and S.A. Doudani, "Locating Man-Made Objects in Low-Resolution Outdoor Scenes," *Applic. of Digital Image Processing*, SPIE Vol. 119, pp. 278-283, 1977.

[Laws77] K.I. Laws, "Circle Detection in Noisy Images," *Semiannual Technical Report*, USCIPI 770, Image Processing Institute, U. of Southern California, Los Angeles, Sep. 1977.

[Lowe80] D.G. Lowe, "Solving for the Parameters of Object Models from Image Descriptions," *Proc. Image Understanding Workshop*, College Park, MD, pp. 121-127, Apr. 1980.

[Lucey76] D.J. Lucey, G.E. Forsen, and M.J. Zoracki, *Processing of FLIR Data on DICIFER*, Pattern Analysis and Recognition Corp., Rome, NY, Report PAR-76-26, 108 pp., July 1976.

[MacVicar-Whelan81] P.J. MacVicar-Whelan and T.O. Binford, "Line Finding with Subpixel Precision," *Proc. Image Understanding Workshop*, Washington, D.C., pp. 26-31, Apr. 1981.

[Martelli76] A. Martelli, "An Application of Heuristic Search Methods to Edge and Contour Detection," *CACM*, Vol. 19, No. 2, pp. 73-83, Feb. 1976.

[Maybeck79] P.S. Maybeck and D.E. Mercier, "A Target Tracker using Spatially Distributed Infrared Measurements," *Proc. 18th IEEE Conf. on Decision and Control*, Fort Lauderdale, FL, pp. 285-9, December 1979. Also pub. in *IEEE Trans. Auto. Control*, Vol. AC-25, No. 2, pp. 222-5, April 1980.

[McIngvale80] P.H. McIngvale and M.C. Pitruzzello, *A Summary of the Development and Evaluation of the Automatic Target Handoff Correlator*, Army Missile Command, Redstone Arsenal, AL, Report RG-80-17, 65 pp., Jan. 1980.

[McKee76] J.W. McKee and J.K. Aggarwal, "Computer Recognition of Partial Views of Three Dimensional Curved Objects," *Proc. 3rd Int. Jnt. Conf. on Pattern Recognition*, Coronado, CA, pp. 499-503, Nov. 1976.

[McQueen81] M.P.C. McQueen, "A Generalization of Template Matching for Recognition of Real Objects," *Pattern Recognition*, Vol. 13, No. 2, pp. 139-145, 1981.

[Merlin75] P.M. Merlin and D.J. Farber, "A Parallel Mechanism for Detecting Curves in Pictures," *IEEE Trans. on Computers*, Vol. C24, pp. 96-98, 1975.

[Milgram78a] D.L. Milgram, A. Rosenfeld, T. Willet, and G. Tisdale, *Algorithms and Hardware Technology for Image Recognition*, Maryland University Computer Science Center, College Park, MD, 287 pp., March 1978.

[Milgram78b] D.L. Milgram, "Edge Point Linking using Convergent Evidence," *Proc. Image Understanding Workshop*, pp. 85-91, Nov. 1978.

[Morris78] R. Morris, "Counting Large Numbers of Events in Small Registers," *Communications of the ACM*, Vol. 21, No. 10, pp. 840-842, Oct. 1978.

[Nahi78] N.E. Nahi and S. Lopez-Mora, "Estimation-Detection of Object Boundaries in Noisy Images," *IEEE Trans. on Automatic Control*, Vol. AC-23, No. 5, pp. 834-846, Oct. 1978.

[Narendra79] P.M. Narendra and J.J. Grabau, *Advanced Pattern Matching Techniques for Autonomous Acquisition*, Honeywell Systems & Research Center, Minneapolis, MN, Report 79SRC104, December 1979, 62 pp.

[Nevatia78] R. Nevatia, "Locating Object Boundaries in Textured Environments," *IEEE Trans. on Computers*, Vol. C-25, No. 11, pp. 1170-1175, Nov. 1976.

[Noges80] E. Noges, A.M. Savol, A.J. Witsmeer, and D. Moerdyke, "Application of an Iterative Feature Matching Algorithm to Terminal Homing," *Image Processing for Missile Guidance*, SPIE Vol. 238, pp. 243-254, 1980.

[O'Gorman73] F. O'Gorman and M.B. Clowes, "Finding Picture Edges through Collinearity of Feature Points," *Proc. 3rd Int. Jnt. Conf. on Artificial Intelligence*, pp. 543-555, 1973.

[O'Rourke81a] J. O'Rourke, "Dynamically Quantized Spaces for Focusing the Hough Transform," *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 737-739, 1981.

[O'Rourke81b] J. O'Rourke, "Motion Detection using Hough Techniques," *IEEE Conf. on Pattern Rec. and Image Processing*, Dallas, pp. 82-87, Aug. 1981.

[Orton79] D.A. Orton and G.N. Yutzi, "The AI2S Microprocessor Based Multimode Tracker," *Wkshp. on Imaging Trackers and Auton. Acq. App. for Missile Guidance*, Redstone Arsenal, AL, pp. 175-91, November 1979.

[Otazo79] J.J. Otazo and R.R. Parenti, "Digital Filters for the Detection of Resolved and Unresolved Targets Embedded in Infrared (IR) Scenes," *Smart Sensors*, Washington, D.C., Proc. SPIE, Vol. 178, pp. 13-24, April 1979.

[Perkins73] W.A. Perkins and T.O. Binford, "A Corner Finder for Visual Feedback," *Computer Graphics and Image Processing*, Vol. 2, pp. 355-376, 1973.

[Perkins76] W.A. Perkins, "Multilevel Vision Recognition System," *Proc. 3rd Int. Jnt. Conf. on Pattern Recognition*, Coronado, CA, pp. 739-744, Nov. 1976.

[Price75] K. Price and R. Reddy, "Change Detection in Multi-Sensor Images," *Proc. 10th Int. Symp. on Remote Sensing of the Environment*, Ann Arbor, MI, pp. 769-776, Oct. 1975.

[Price81] K.E. Price, "Relaxation Matching Applied to Aerial Images," *Proc. Image Understanding Workshop*, Washington, D.C., pp. 22-25, Apr. 1981.

[Pridgen79] J.H. Pridgen, W.W. Boyd, W.C. Choate, E.E. Mooty, and R.B. Ottesen, "Terminal Homing Applications of Solid-State Imaging Devices (THASSID) Composite Tracking Concepts," *Digital Processing of Aerial Images*, Huntsville, AL, Proc. SPIE, Vol. 186, pp. 73-87, May 1979.

[Quam80] L.H. Quam, "A Storage Representation for Efficient Access to Large Multi-Dimensional Arrays," *Proc. Image Understanding Workshop*, College Park, MD, pp. 104-111, Apr. 1980.

[Reid79] D.B. Reid, *The Application of Multiple Target Tracking Theory to Ocean Surveillance*, Lockheed Palo Alto Research Lab., Palo Alto, CA, 1979.

[Rosenfeld69] A. Rosenfeld, *Picture Processing by Computer*. Academic Press, New York, 1969.

[Rosenfeld78] A. Rosenfeld, "Iterative Methods in Image Analysis," *Pattern Recognition*, Vol. 10, No. 3, pp. 181-7, 1978.

[Sabbah81] D. Sabbah, "Design of a Highly Parallel Visual Recognition System", *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 722-727, 1981.

[Shapiro74] S.D. Shapiro, "Detection of Lines in Noisy Pictures using Clustering," *2nd Int. Jnt. Conf. on Pattern Recognition*, Copenhagen, pp. 317-318, Aug. 1974.

[Shapiro75] S.D. Shapiro, "Transformations for the Computer Detection of Curves in Noisy Pictures," *Computer Graphics and Image Processing*, Vol. 4, pp. 328-338, 1975.

[Shapiro76] S.D. Shapiro, "An Extension of the Transform Method of Curve Detection for Textured Image Data," *3nd Int. Jnt. Conf. on Pattern Recognition*, Coronado, CA, pp. 205-207, Nov. 1976. See also *IEEE Trans. on Computers*, Vol. C-27, p. 254, Mar. 1978.

[Shapiro78a] S.D. Shapiro, "Feature Space Transforms for Curve Detection," *Pattern Recognition*, Vol. 10, pp. 129-143, 1978.

[Shapiro78b] S.D. Shapiro, "Generalization of the Hough Transform for Curve Detection in Noisy Pictures," *Proc. 4th Int. Jnt. Conf. on Pattern Recognition*, Kyoto, Japan, pp. 710-714, Nov. 1978.

[Shapiro78c] S.D. Shapiro, "Curve Formation Using Transforms for Pictures Governed by Differential Equations," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 10, pp. 763-765, Oct. 1978.

[Shapiro79a] S.D. Shapiro and A. Iannino, "Geometric Constructions for Predicting Hough Transform Performance," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 3, July 1979.

[Shapiro79b] S.D. Shapiro, "Use of the Hough Transform for Image Data Compression," *IEEE Conf. on Pattern Rec. and Image Processing*, Chicago, pp. 576-582, Aug. 1979.

[Sklansky78] J. Sklansky, "On the Hough Technique for Curve Detection," *IEEE Trans. on Computers*, Vol. C27, No. 10, pp. 923-926, Oct. 1978.

[Sloan80] K.R. Sloan, Jr., and D.H. Ballard, "Experience with the Generalized Hough Transform," *Proc. DARPA Image Understanding Wkshp.*, pp. 150-156, Apr. 1980. Also pub. as U. of Rochester Computer Science Department TR-83, October 1980.

[Sloan81] K.R. Sloan, Jr., "Dynamically Quantized Pyramids," *Proc. 7th Int. Jnt. Conf. on Artificial Intelligence (IJCAI-81)*, Vol. 2, pp. 734-738, 1981.

[Stockman77] G.C. Stockman and A.K. Agrawala, "Equivalence of Hough Curve Detection to Template Matching," *CACM*, Vol. 20, No. 11, pp. 820-822, Nov. 1977.

[Tanimoto77] S.L. Tanimoto and T. Pavlidis, "The Editing of Picture Segmentations Using Local Analysis of Graphs," *CACM*, Vol. 20, No. 4, pp. 223-229, Apr. 1977.

[Tisdale77] G.E. Tisdale, "A Digital Image Processor for Automatic Target Cueing, Navigation, and Change Detection," *Airborne Reconnaissance – Tactical/Real Time*, Reston, VA, Proc. SPIE, Vol. 101, pp. 112-9, April 1977.

[Tisdale79] G.E. Tisdale, A.R. Helland, and J. Shipley, *Automatic Target Cueing (AUTO-Q) System Engineering Model Design*, Westinghouse Systems Development Division, Baltimore, MD, Report 80-0488, September 1979, 60 pp.

[Touchberry77] R. Touchberry and R. Larson, "Symbolic Analysis of Images using Prototype Similarity," *Proc. Image Understanding Workshop*, pp. 78-82, Apr. 1977.

[Tsuji77] S. Tsuji and F. Matsumoto, "Detection of Elliptic and Linear Edges by Searching Two Parameter Spaces," *Proc. 5th Int. Jnt. Conf. on Artificial Intelligence*, Cambridge, MA, pp. 700-705, Aug. 1977.

[Tsuji78] S. Tsuji and F. Matsumoto, "Detection of Ellipses by a Modified Hough Transformation," *IEEE Trans. on Computers*, Vol. C-27, No. 8, pp. 777-781, Aug. 1978.

[van Veen81] T.M. van Veen and F.C.A. Groen, "Discretization Errors in the Hough Transform," *Pattern Recognition*, Vol. 14, pp. 137-145, 1981.

[WDESC78] *Intelligent Tracking Techniques*, Westinghouse Defense and Elec. Sys. Center, Systems Development Division, Baltimore, MD, Report 78-0503, December 1978, 75 pp.

[Wechsler77] H. Wechsler and J. Sklansky, "Finding the Rib Cage in Chest Radiographs," *Pattern Recognition*, Vol. 9, No. 2, pp. 21-30.

[Winkler78] G. Winkler and K. Vattrodt, "Measures for Conspicuousness," *Computer Graphics and Image Processing*, Vol. 8, pp. 355-368, 1978.

[Woolfson78] M.G. Woolfson, "Digital Area Correlation Tracker," *Proc. IEEE 1978 National Aerosp. and Electron. Conf. (NAECON 78)*, Dayton, OH, pp. 882-6, May 1978.

[Yam81] S. Yam and L.S. Davis, "Image Registration using Generalized Hough Transforms," *IEEE Conf. on Pattern Rec. and Image Processing*, Dallas, pp. 526-533, Aug. 1981.

[Zabele79] G.S. Zabele, *On Optimum Line Detection in Noisy Images using the Hough Transform*. Master's Thesis, Clarkson College, Jan. 1979.